Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

1. (original) A network management system for managing a plurality of network devices, comprising:

a device database for storing a native configuration for one of the plurality of network devices;

a component database for storing configuration information used to configure the plurality of network devices, wherein the configuration information is stored as a plurality of components and a plurality of candidate components;

a device learning module for receiving the native configuration from the device database, identifying the configuration information from the native configuration, and storing the configuration information in the component database; and

a grammar builder for receiving a candidate component, resolving the candidate component into a component, and storing the component in the component database.

2. (original) The system of claim 1, wherein the configuration information identified from the native configuration information comprises a policy.
3. (original) The system of claim 2, wherein the policy comprises a plurality of components.
4. (original) The system of claim 2, wherein the policy comprises a template adapted to aggregate together a plurality of components all sharing a common attribute.
5. (original) The system of claim 4, wherein the common attribute comprises membership in a network configuration function for one of the plurality of network devices.
6. (original) The system of claim 1, wherein the plurality of components comprise a policy-driven configuration for one of the plurality of network devices.
7. (original) The system of claim 1, wherein each component comprises a component syntax block.
8. (original) The system of claim 1, wherein each component comprises a configuration command for a configuration command language, the configuration command adapted to configure a function on the network device.

9. (original) The system of claim 1, wherein the configuration information identified from the native configuration information comprises a candidate component.
10. (original) The system of claim 1, wherein the grammar builder receives the candidate component from the device learning module.
11. (original) The system of claim 1, wherein the grammar builder receives the candidate component from the component database.
12. (original) The system of claim 1, wherein the device learning module further comprises a lexer adapted to tokenize the native configuration, and a parser adapted to parse the tokenized native configuration to identify a plurality of components and a candidate component within the tokenized native configuration.
13. (original) The system of claim 12, wherein the parser is configured to identify the plurality of components and the candidate component by comparing the tokenized native configuration to the configuration information stored in the component database.
14. (original) The system of claim 13, wherein the parser is adapted to identify the candidate component by detecting a portion of the tokenized native configuration that does not match the configuration information stored in the component database.
15. (original) The system of claim 13, wherein the parser is adapted to identify the plurality of components by detecting portions of the tokenized native configuration that match the configuration information stored in the component database.
16. (original) The system of claim 12, wherein the lexer is configured to tokenize the native configuration according to a grammar embodied in the configuration information stored in the component database.
17. (original) The system of claim 12, wherein at least a first portion of the lexer and the parser are recompiled each time the device learning module receives a native configuration.
18. (original) The system of claim 17, wherein a second portion of the lexer and the parser remains constant across multiple receptions of native configurations.
19. (original) The system of claim 12, wherein the device learning module further comprises a policy matcher, adapted to parse the plurality of components and recognize a policy contained in the plurality of components.

20. (original) The system of claim 19, wherein the device learning module is adapted to output a policy-driven configuration and zero or more candidate components.

21. (original) The system of claim 1, wherein the device learning module is adapted to compile a policy-driven configuration into a native configuration.

22. (original) The system of claim 1, further comprising a device data storage for storing device-specific information about the plurality of network devices.

23. (original) The system of claim 1, wherein the grammar builder is adapted to retrieve command root completion information from one of the plurality of network devices, create a plurality of command root derived components from the command root completion information, and resolve the candidate component into a component by comparing the candidate component to the plurality of command root derived components, to identify one of the plurality of command root derived components that matches the candidate component.

24. (original) A method of parsing a native configuration into a policy-driven configuration, comprising:

receiving configuration information comprising a plurality of components;

receiving the native configuration;

tokenizing the native configuration using a lexer module;

parsing the tokenized native configuration using a parser module, to identify a plurality of input components contained in the native configuration and match the plurality of input components with the plurality of components;

parsing the tokenized native configuration using the parser module, to identify one or more unknown regions contained in the native configuration, which do not match any of the plurality of components;

emitting a tree of components, comprising the plurality of matched input components and the one or more unknown regions;

processing the one or more unknown regions to identify one or more candidate components;

analyzing the tree of components to identify one or more policies present in the tree of components; and

outputting the one or more policies and the one or more candidate components, as a policy-driven configuration.

25. (original) The method of claim 24, further comprising:

compiling the lexer module using the configuration information, such that the lexer module is adapted to tokenize the native configuration according to a grammar embodied in the configuration information;

compiling the parser module using the configuration information, such that the parser module is adapted to match the tokenized native configuration to the configuration information;

26. (original) The method of claim 24, further comprising reorganizing the tree of components to enhance user comprehension of the tree.

27. (original) The method of claim 24, further comprising compressing the tree to remove unnecessary data from the tree.

28. (original) The method of claim 27, wherein the unnecessary data comprises an empty node in the tree.

29. (original) The method of claim 24, wherein the plurality of components comprises a policy, and analyzing the tree of components comprises:

retrieving the policy;

comparing each of the plurality of components in the policy with the plurality of matched input components contained in the tree of components;

aborting the analysis if any of the plurality of components in the policy is not found in the plurality of matched input components; and

identifying the policy as present in the tree of components if all of the plurality of components in the policy are found in the plurality of matched input components.

30. (original) The method of claim 29, further comprising removing all of the components in the plurality of matched input components that were matched to the plurality of components in the policy, and inserting the policy into the tree of components.

31. (original) The method of claim 29, wherein the policy comprises an ordered policy, and identifying the policy further comprises identifying the ordered policy as present in the tree of components if all of the plurality of components in the ordered policy are found in the plurality of matched input components in the same order as in the ordered policy.

32. (original) The method of claim 24, wherein parsing the tokenized native configuration to identify one or more unknown regions comprises:

detecting a parse error, wherein the parser fails to recognize a portion of the tokenized native configuration;

marking the portion for further processing; and

continuing to parse the tokenized native configuration.

33. (original) The method of claim 24, wherein processing the one or more unknown regions to identify one or more candidate components comprises:

walking the tree of components;

marking each of the plurality of matched input components in the tree of components;

re-walking the tree of components;

identifying a token within one of the one or more unknown regions within the tree of components.

walking the tree of components in each of two directions, beginning at the identified token, to identify a beginning point and an end point of the unknown region; and

marking the unknown region as a candidate component.

34. (new) The system of claim 1, wherein the grammar builder comprises:

a syntax tree acquisition module, configured to receive a command root and output a syntax tree for the command root; and

a syntax tree transformation module, configured to receive the syntax tree and output a grammar based on the syntax tree.

35. (new) The system of claim 34, wherein the syntax tree acquisition module comprises a command root completion module.

36. (new) The system of claim 35, wherein the command root completion module comprises a command help walker.

37. (new) The system of claim 34, wherein the syntax tree transformation module is at least partially vendor-neutral.

38. (new) The system of claim 34, wherein the syntax tree transformation module comprises a syntax tree post-processor.

39. (new) The system of claim 34, further comprising a semantic effect examination module, configured to identify one or more semantic effects triggered by adding or subtracting a component of the grammar, and to catalog the one or more semantic effects in the grammar.
40. (new) The system of claim 39, wherein the semantic effect examination module is configured to catalog the one or more semantic effects by annotating the component that triggers the semantic effect.
41. (new) The system of claim 39, wherein the one or more semantic effects comprise one or more of disappearing default values, equivalent format transformations, linked command appearance, linked command disappearance and unique command parameters.
42. (new) The system of claim 34, wherein the grammar is output in human-readable form.
43. (new) The system of claim 34, wherein the grammar builder is configured to accept a plug-in designed to handle specialized processing needs
44. (new) The system of claim 43, wherein specialized processing needs comprise loop processing.
45. (new) The system of claim 34, wherein the grammar builder is configured to construct both a positive grammar and a negative grammar for the command root.
46. (new) A method of producing a grammar, comprising:

receiving unknown command information;

generating an abstract syntax tree for the unknown command information; and

transforming the abstract syntax tree into a component tree.
47. (new) The method of claim 46, wherein the unknown command information comprises a command root.
48. (new) The method of claim 46, wherein generating an abstract syntax tree comprises walking a command completion feature for the unknown command information.
49. (new) The method of claim 46, further comprising examining the component tree for semantic effects.
50. (new) The method of claim 49, wherein examining the component tree comprises modifying the component tree; applying the modified component tree to a network device; retrieving a native configuration from the network device; and determining whether the native configuration matches the modified component tree.

51. (new) A method of generating a syntax tree for a command for a network device, comprising:

identifying a command root;

providing the command root to the network device;

using a command completion feature on the network device for the command root, to identify a plurality of next completions for the command root;

recursively entering each of the plurality of next completions;

recording in the syntax tree each next completion; and

outputting the syntax tree.

52. (new) The method of claim 51, wherein the command root is identified by receiving a candidate component containing the command root.

53. (new) The method of claim 51, further comprising terminating the recursion step for a branch when a termination character is encountered in the command completion feature.

54. (new) The method of claim 53, wherein the termination character comprises an EOL character.

55. (new) The method of claim 51, wherein entering each of the plurality of next completions further comprises entering a proxy value for a placeholder next completion.

56. (new) The method of claim 51, wherein recording in the syntax tree each next completion further comprises recording metadata about the next completion.

57. (new) The method of claim 51, further comprising handling special processing needs.

58. (new) The method of claim 57, wherein the special processing needs comprise loop processing.

59. (new) A method of transforming a syntax tree including a plurality of syntax nodes into a grammar, comprising:

transforming the plurality of syntax nodes into a plurality of equivalent grammar constructs, resulting in the creation of a grammar;

transforming the grammar to remove unnecessary terminations; and

re-factoring the grammar to remove structural anomalies.

60. (new) The method of claim 59, wherein the transforming step is performed using a plurality of translation rules.

61. (new) The method of claim 59, wherein the syntax tree comprises a plurality of commands and a plurality of termination characters, and wherein transforming the grammar to remove unnecessary terminations comprises ensuring that each command follows a single path in the syntax tree to a single termination character.

62. (new) The method of claim 59, wherein the structural anomalies comprise one or more of common endings, common sub-expressions, orphaned expressions, and repeated segments.

63. (new) The method of claim 59, wherein the syntax tree comprises a command, further comprising:

identifying a plurality of boundaries of the command; and

inserting boundary markers denoting the beginning and end of the command.

64. (new) The method of claim 63, wherein identifying a plurality of boundaries of the command comprises:

receiving a positive grammar for a positive form of the command, the positive grammar comprising a plurality of positive nodes;

receiving a negative grammar for a negative form of the command, the negative grammar comprising a plurality of negative nodes;

walking the plurality of positive nodes;

comparing each of the plurality of positive nodes with a negative node in a corresponding position within the negative grammar;

identifying the two compared nodes as one of the plurality of boundaries for the command if the two compared nodes each comprise a termination character;

marking the plurality of positive nodes walked as being the positive form of the command;

marking the plurality of negative nodes corresponding to the walked plurality of positive nodes as being the negative form of the command; and

merging the marked plurality of positive nodes and marked plurality of negative nodes.

65. (new) The method of claim 64, further comprising walking the plurality of negative nodes.

66. (new) A method of identifying a semantic effect caused by modification of a device configuration on a network device, comprising:

selecting the network device;

retrieving the device configuration from the network device;

storing the device configuration;

modifying the device configuration on the network device;

retrieving the modified device configuration from the network device;

comparing the modified device configuration with the stored device configuration;

identifying a difference between the modified and stored device configurations; and

processing the difference to identify the semantic effect.

67. (new) The method of claim 66, wherein modifying the device configuration comprises adding a command to the device configuration.

68. (new) The method of claim 66, wherein modifying the device configuration comprises removing a command from the device configuration.

69. (new) The method of claim 66, further comprising parsing the stored and modified device configurations into component trees.

70. (new) The method of claim 66, wherein the semantic effect comprises one or more of disappearing default values, equivalent format transformations, linked command appearance, linked command disappearance and unique command parameters.

71. (new) The method of claim 66, further comprising determining an attribute of the device configuration to modify.

72. (new) The method of claim 66, wherein processing the difference comprises identifying a default value.

73. (new) The method of claim 66, wherein processing the difference comprises identifying an equivalent format.

74. (new) The method of claim 67, wherein the device configuration comprises a first instance of a command having a first value for an attribute of the command, the modification comprises adding a second instance of the command, the second instance of the command having a second value for the attribute, and processing the difference comprises:

identifying a number of instances of the command present in the modified device configuration; and

if the number of instances is two, marking the attribute of the command as causing the command to be unique.

75. (new) The method of claim 74, further comprising, if the number of instances is one, marking the attribute of the command as causing the command to be non-unique.

76. (new) The method of claim 67, wherein the difference comprises a second command present in the modified device configuration but not the stored device configuration, and processing the difference to identify the semantic effect comprises recording the second command.

77. (new) The method of claim 67, wherein the difference comprises a second command present in the stored device configuration but not the modified device configuration, and processing the difference to identify the semantic effect comprises recording the second command.

78. (new) The method of claim 68, wherein the difference comprises a second command present in the modified device configuration but not the stored device configuration, and processing the difference to identify the semantic effect comprises recording the second command.

79. (new) The method of claim 68, wherein the difference comprises a second command present in the stored device configuration but not the modified device configuration, and processing the difference to identify the semantic effect comprises recording the second command.

80. (new) A method of compiling a native device configuration for a network device, comprising:

retrieving a full grammar for the network device;

retrieving a policy-driven configuration for the network device;

configuring a parser using the full grammar;

supplying the policy-driven configuration to the parser;

recursively walking the policy-driven configuration to generate the native device configuration.

81. (new) The method of claim 80, wherein a recursion of the recursively walking step is aborted if an incomplete output string is generated for the recursion.

82. (new) The method of claim 80, wherein the policy-driven configuration comprises an incremental configuration.

83. (new) A method of auditing a native configuration running on a network device, comprising:

retrieving the running configuration from the network device;

retrieving a stored configuration corresponding to the running configuration;

comparing the running configuration and the stored configuration; and

recording any differences between the running configuration and the stored configuration.

84. (new) The method of claim 83, wherein the stored configuration comprises an instance tree for a policy-driven configuration, wherein comparing the running and stored configurations comprises comparing the instance tree with a second instance tree, further comprising parsing the running configuration to generate the second instance tree.

85. (new) The method of claim 83, wherein the differences comprise one or more missing policy linkages in the retrieved configuration.

86. (new) The method of claim 83, wherein the differences comprise one or more missing components in one of the retrieved or stored configurations.